Project: 총알 피하기 UPGRADE

소프트웨어학부

20192000 최유찬

* 미션 구현여부 표

미션1	미션2	미션3	미션4	미션5	미션6	미션7	미션8	미션9	미션10	미션11
O	O	O	O	O	O	O	O	O	O	O

미션1: 비행기가 총알에 맞았을 때 효과음이 발생하도록 한다.

```
8   def collision(obj1, obj2):
9       if math.sqrt((obj1.pos[0] - obj2.pos[0]) ** 2 + (obj1.pos[1] - obj2.pos[1]) ** 2) < 20:
10          collision_sound.play() #부딪히면 효과음(미션1)
11          return True
12      return False
```

```
51  #부딪혔을 때 효과음 선언
52  collision_sound = pygame.mixer.Sound('collision.wav')
```

- 다음과 같이 제가 가지고 있는 collision.wav 파일을 collsion_sound라는 전역변수로 선언하였습니다. 그리고 def collision라는 함수안에서 만약 비행기가 총알을 맞는 경우에 해당하는 if문안에 collision_sound를 재생시키고 난 뒤에 True를 리턴하도록 코드를 수정하였습니다.

미션2: 비행기가 총알에 맞았을 때 터지는 그림 효과가 나타나도록 한다.

```
8   def collision(obj1, obj2):
9       if math.sqrt((obj1.pos[0] - obj2.pos[0]) ** 2 + (obj1.pos[1] - obj2.pos[1]) ** 2) < 20:
10          collision_sound.play() #부딪히면 효과음(미션1)
11          screen.blit(boom_image, (obj1.pos[0], obj1.pos[1])) #부딪혔을 때 터지는 그림효과(미션2)
12          pygame.display.update() #터지는 그림을 그리고 화면 갱신
13          return True
14      return False
```

```
53  #부딪혔을 때 효과음 선언
54  collision_sound = pygame.mixer.Sound('collision.wav')
55  #부딪혔을 때 터지는 그림
56  boom_image = pygame.image.load('boom.png')
57  boom_image = pygame.transform.scale(boom_image, (64, 64)) #비행기와 같은 크기로
58
```

- 터지는 효과의 그림을 boom_image라는 변수로 선언하고 57번째 줄의 코드를 통해 터지는 효과의 그림을 비행기와 같은 크기로 조절하였습니다. 그리고 def collision 함수 안에서 비행기가 충

돌을 감지하였을 때 터지는 그림을 비행기의 위치에 맞게 나타나도록 하였습니다. 처음에 시도할 때 그림이 안나오서 왜 그런지 고민을 했었는데, 수업 시간에 교수님께서 그림을 그리고 덮는 과정이 필요하다고 했던 것이 생각났습니다. 그래서 12번 째 줄의 코드와 같이 터지는 화면을 그리고 난 후 화면을 갱신하도록 하는 코드를 추가하여 문제를 해결하였습니다.

미션3: 배경그림이 비행기의 움직임에 반응하여 같이 움직이도록 한다.

```
(2053, 1500)
(2053, 1500)
(2053, 1500)
(2053, 1500)
(2053, 1500)
(2053, 1500)
(2053, 1500)
(2053, 1500)
(2053, 1500)
(2053, 1500)
```

```
73
74    bg_posx = -500 #배경을 그리기 위한 x좌표
75    bg_posy = -100 #y좌표
76
```

```
116
117     if event.type == pygame.KEYDOWN:
118         if -1053< bg_posx <0 and -700< bg_posy < 0:    #배경이미지의 끝에 도달하기 전까지만 움직이도록 한다
119             if event.key == pygame.K_LEFT:      #미션 3을 수행하기 위해 아래와 같이 4가지 경우의 형태로
120                 bg_posx += -0.05 * dt
121             elif event.key == pygame.K_RIGHT:
122                 bg_posx += 0.05 * dt
123             elif event.key == pygame.K_UP:
124                 bg_posy += 0.05 * dt
125             elif event.key == pygame.K_DOWN:
126                 bg_posy += -0.05 * dt
127         if bg_posx <= -1053:    #127~134줄의 코드는 끝에 도달하였을 때
128             bg_posx += 1        #배경의 x,y좌표를 다시 118 if문에 해당하는 범위 안으로 넣어주기 위해서이다.
129         elif bg_posx >= 0:      #이 작업을 해주지 않으면 화면 끝에 도달 후, 화면이 다시 움직이지 않는다.
130             bg_posx -= 1
131         if bg_posy <= -700:
132             bg_posy += 1
133         elif bg_posy >= 0:
134             bg_posy -=1
135
```

```
135
136         screen.blit(bg_image, (bg_posx, bg_posy)) # 배경 그리기
137         player.update(dt, screen)
138         player.draw(screen)
139
```

Bg_image라는 배경이미지를 그리기 위한 x,y좌표를 각각 bg_posx와 bg_posy로 두었습니다. 초기에 -500과 -100을 한 것은 시작 위치를 조정하기 위함입니다. while running이 실행되는 동안, 입력받은 방향키를 통해 배경을 방향키와 같은 방향으로 설정하도록 작업하였습니다. 왼쪽 오른쪽은 bg_posx로, 위 아래는 bg_posy로 구분할 수 있게 하였습니다. 그리고 screen.blit명령어를 이용해 배경사진을 그렸습니다. 그리고 배경사진이 끝에 도달하면 더 이상 움직이지 않기 하기 위해 배경이미지의 크기가 bg_image.get_rect().size 를 통해 (2053,1500)임을 확인하였습니다. 118줄의 if 문에서 숫자의 의미는 -1053 = 스크린의 가로길이(1000) – 이미지의 가로길이(2053) 이고,

-700 = 스크린의 세로길이(800) – 이미지의 세로길이(1500) 입니다. 그리고 최대값은 각각 0입니다. 127~134의 코드는 배경이미지의 끝에 도달하였을 때, 해당 값을 다시 if문을 만족하는 값으로 넣어주어 화면이 계속 움직이게 하기 위함입니다. 만약 이 코드가 없다면 화면 끝에 도달하였을 때 화면이 다시 움직이지 않는 문제가 발생합니다.

..

미션4: 비행기가 총알과 여러 번 충돌해야 게임이 종료되도록 한다. (생명력 개념 도입)

```
8   lastcollision = None #초기값
9   def collision(obj1, obj2):
10      global lastcollision #lastcollision 변수 선언
11      if lastcollision is not None and time.time() - lastcollision < 3: # 마지막 충돌 시간과 현재 시간을
12          return False
13      if math.sqrt((obj1.pos[0] - obj2.pos[0]) ** 2 + (obj1.pos[1] - obj2.pos[1]) ** 2) < 20:
14          collision_sound.play() #부딪히면 효과음(미션1)
15          screen.blit(boom_image, (obj1.pos[0], obj1.pos[1])) #부딪혔을 때 터지는 그림효과(미션2)
16          pygame.display.update() #터지는 그림을 그리고 화면 갱신
17          lastcollision = time.time() #무적시간이 아니면서 충돌을 했다면 마지막 충돌 시간을 업데이트
18          return True
19      return False
```

```
: # 마지막 충돌 시간과 현재 시간을 비교해서 그 차이가 무적 시간보다 작다면 충돌을 감지하지 못하도록
```

- 함수에서 마지막 충돌 시간과 현재 시간을 비교해서 그 차이가 무적 시간보다 작다면 return False를 하도록 하고, 무적시간이 아니면서 충돌을 했다면 마지막 충돌 시간을 업데이트 해주는 식으로 해결하였습니다. 8번째 줄 코드에서 처음 이전에는 충돌이 발생하지 않았으니 None으로 초기값을 주었고, 10~12번째 줄의 코드에서 무적상태를 구현하였습니다. 현재시간에서 마지막충돌 시간이 3초(무적 시간)보다 작으면 충돌을 감지하지 않도록 하였습니다. 그리고 무적시간이 아니면서 충돌을 했다면 마지막 충돌시간을 업데이트하도록 17번째 줄의 코드를 작성하였습니다. 2번째 사진은 11번 코드의 주석내용입니다.

미션5: 무적시간동안 비행기가 반짝거리도록 한다.

```
player.py > Player > draw
1   import math
2   import pygame
3
4   class Player:
5       nodamgetime = False #비행기를 깜빡이게 하는 시간 체크하기 위한 bool 변수
6
```

```
26          if self.nodamgetime == False:   #이 변수가 False 일때만 출력하게 하면 True 일때는 출력이 안되므로 비
27              rotated = pygame.transform.rotate(self.image, self.angle) #회전
28              calib_pos = (self.pos[0] - rotated.get_width()/2,
29                           self.pos[1] - rotated.get_height()/2)
30              screen.blit(rotated, calib_pos)
31
```

```
#이 변수가 False 일때만 출력하게 하면 True 일때는 출력이 안되므로 비행기를 깜박이게 구현할 수 있다.
```

- 비행기를 반짝이게 하기 위해 제가 생각한 방법은 첫번째 사진과 같이 Player 클래스에 nodamgetime이라는 bool변수를 맴버 변수로 설정하고, draw 함수에서 그림을 그릴 때 nodamgetime이 False일때만 그리도록 하는 방법을 생각했습니다. 그러면 main.py에서 반짝이게 하고 싶은 시간동안 True 값을 할당하면 반짝이는 것을 구현할 수 있다고 생각하였습니다.

```
10  lastcollision = None #초기값
11  def collision(obj1, obj2):
12      global lastcollision #lastcollision 변수 선언
13      if lastcollision is not None and time.time() - lastcollision < 3:# 마지막 충돌 시간과 현재 시간을 비교해
14          if ((int(((time.time()-lastcollision))*10))%2 == 0):    #일정시간동안 깜빡이게 하기 위해 0과 1의 값이
15              player.nodamgetime = True   #위 조건을 만족할때 True를 넣어주면 이 시간에는 비행기가 출력이 안된다
16          else:
17              player.nodamgetime = False  #위 조건을 만족하지 않을 때는 False값을 넣어줘서 비행기가 출력된다.
18
```

- 다음과 같이 작성하였습니다. 14번 줄의 코드를 print해보면 0과 1이 계속 번갈아가면서 출력됩니다. 이를 이용하여 이 값이 0 일때는 True를 전달하고, 1일때는 False를 전달하여 비행기가 깜빡거리도록 구현하였습니다.

```
147     if gameover:
148         player.nodamgetime = False  #게임이 종료되었을 때 비행기가 사라지지 않게 하기 위해
```

- 위의 작업만 하고 끝냈을 때, 비행기의 생명력이 다 떨어져 게임이 종료될 때 비행기가 사라져 보이지 않는 문제가 발생하였습니다. 따라서 다음과 같이 gameover가 되면 nodamgetime을 False로 주도록 하여 비행기가 계속 보이도록 해주었습니다.

미션6: 남은 생명력을 막대기(5점)와 숫자(5점)로 표시한다.

```
118     if gameover:
119         draw_text("GAME OVER", 100, (WIDTH/2 - 300, HEIGHT/2 - 50), (255,255,255))
120         txt = "Time: {:.1f}  Bullets: {}".format(score, len(bullets))
121         draw_text(txt, 32, (WIDTH/2 - 150, HEIGHT/2 + 50), (255,255,255))
122     else:
123         score = time.time() - start_time
124         txt = "Time: {:.1f}  Bullets: {}".format(score, len(bullets))
125         draw_text(txt, 32, (10, 10), (255,255,255))
126         lifetxt = "life: %d" % (player.life)
127         draw_text(lifetxt, 32, (WIDTH/2, 10), (255,255,255)) #게임 상단에 life를 정수로 표현
128
129         for life in range(player.life):
130             screen.blit(life_image, (life*30 + 600 ,10)) #목숨 개수 만큼 오른쪽 상단에 life이미지 출력
131
132     pygame.display.update() #화면에 새로운 그림을 그린다 (화면을 갱신한다)
133
```

```
68  #목숨을 표현하기 위한 하트
69  life_image = pygame.image.load('heart.png')
70  life_image = pygame.transform.scale(life_image, (30, 30))
```

- 목숨 개수만큼 하트이미지로 표시하기 위해 2번째 사진에서 보듯이 life_image라는 변수에 하트

이미지를 불러오고 적당한 크기로 설정하였습니다. 그리고 129~130줄의 코드에서 생명력의 개수만큼 하트이미지를 일정한 간격으로 나타나도록 하였습니다. 그리고 126~127줄에 코드에서 생명력이 얼마나 남았는지를 게임 상단에 정수로 표시하도록 하였습니다. 결과는 다음과 같이 나왔습니다.


```
4   class Player:
5       def __init__(self, x, y):
6           self.image = pygame.image.load('player.png')
7           self.image = pygame.transform.scale(self.image, (64, 64))
8           self.pos = [x, y]
9           self.to = [0, 0] # 가르키고 있는 방향
10          self.acc = [0, 0]
11          self.angle = 0
12          self.life = 7 #생명력
```

미션7: 총알을 여러 종류로 만들고, 종류별로 크기와 색깔을 다르게 표현한다.

- 저는 총알의 종류를 3가지로 만들었습니다. 총알 1에서 총알 3으로 갈수록 점점 커지도록 구현하였습니다.

```
19  class Bullet2:  #총알2
20      def __init__(self, x, y, to_x, to_y):
21          self.pos = [x, y]
22          self.to = [to_x, to_y]
23          self.radius = 13 #총알1보다 크기 좀 더 크게
24          self.color = [0,200,0] #초록색 총알
25
26      def update_and_draw(self, dt, screen):
27          width, height = screen.get_size()
28          self.pos[0] = (self.pos[0] + dt*self.to[0]) % width
29          self.pos[1] = (self.pos[1] + dt*self.to[1]) % height
30          pos_int = (int(self.pos[0]), int(self.pos[1]))
31          pygame.draw.circle(screen, self.color, pos_int, self.radius)
32
```

```
33  class Bullet3:
34      def __init__(self, x, y, to_x, to_y):
35          self.pos = [x, y]
36          self.to = [to_x, to_y]
37          self.radius = 30 #총알2보다도 더 크게
38          self.color = [0,0,250] #파란색 총알
39
40      def update_and_draw(self, dt, screen):
41          width, height = screen.get_size()
42          self.pos[0] = (self.pos[0] + dt*self.to[0]) % width
43          self.pos[1] = (self.pos[1] + dt*self.to[1]) % height
44          pos_int = (int(self.pos[0]), int(self.pos[1]))
45          pygame.draw.circle(screen, self.color, pos_int, self.radius)
46
```

- 우선 bullet.py에 총알1을 구현한 것처럼 총알2와 총알3의 클래스를 추가하였습니다.

```
import time
import random as rnd
import math
import pygame
from player import Player
from bullet import Bullet1
from bullet import Bullet2
from bullet import Bullet3
```

- 그리고 main.py에 모든 총알을 사용할 수 있도록 import하였습니다.

```
bullets1 = [] #총알1
for i in range(3):
    bullets1.append(Bullet1(0, rnd.random()*HEIGHT, rnd.random()-0.5, rnd.random()-0.5))

bullets2 = [] #총알2
for i in range(3):
    bullets2.append(Bullet2(0, rnd.random()*HEIGHT, rnd.random()-0.5, rnd.random()-0.5))

bullets3 = [] #총알3
for i in range(3):
    bullets3.append(Bullet3(0, rnd.random()*HEIGHT, rnd.random()-0.5, rnd.random()-0.5))
```

```
        time_for_adding_bullets += dt*0.1 #밸런스 조정을 위해 총알개수가 늘어나는 시간 감소
        if time_for_adding_bullets > 1000:
            bullets1.append(Bullet1(0, rnd.random()*HEIGHT, rnd.random()-0.5, rnd.random()-0.5))
            bullets2.append(Bullet2(0, rnd.random()*HEIGHT, rnd.random()-0.5, rnd.random()-0.5))
            bullets3.append(Bullet2(0, rnd.random()*HEIGHT, rnd.random()-0.5, rnd.random()-0.5))
            time_for_adding_bullets -= 1000
```

- 기존에 있던 코드에 다음과 같이 총알1,2,3이 랜덤으로 추가되는 것과, 개수가 추가되는 부분을 추가하였습니다.

```
    for b in bullets1: #총알1
        b.update_and_draw(dt, screen)
    for b in bullets2: #총알2
        b.update_and_draw(dt, screen)
    for b in bullets3: #총알3
        b.update_and_draw(dt, screen)
```

- 그리고 화면에 총알이 표시되도록 하였습니다.

```
    else:
        score = time.time() - start_time
        txt = "Time: {:.1f}  Bullets: {}".format(score, len(bullets1) + len(bullets2) + len(bullets3))#
```

```
    if gameover:
        player.nodamgetime = False  #게임이 종료되었을 때 비행기가 사라지지 않게 하기 위해
        draw_text("GAME OVER", 100, (WIDTH/2 - 300, HEIGHT/2 - 50), (255,255,255))
        txt = "Time: {:.1f}  Bullets: {}".format(score, len(bullets1)+ len(bullets2) + len(bullets3))#
```

- 그리고 점수판에 총알 개수를 세어주는 코드에 총알1,2,3의 개수가 저장되도록 다음과 같이 수정하였습니다.

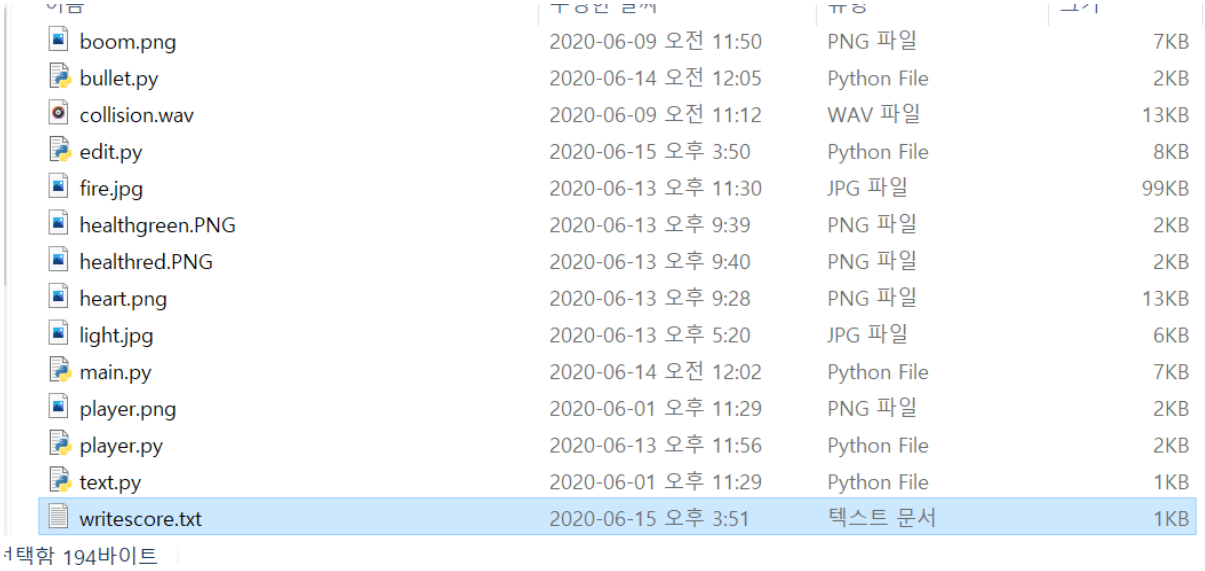
미션8: 총알을 종류마다 플레이어와 충돌했을 때 차감되는 생명력을 다르게 한다.

```
181     if not gameover:
182         for b1 in bullets1: #총알1
183             if collision(player, b1):
184                 player.life -= 1 #생명력을 1감소
185                 if player.life <= 0: #생명력이 0보다 작으면 게임을 종료하도록
186                     gameover = True
187         for b2 in bullets2: #총알2
188             if collision(player, b2):
189                 player.life -= 2 #생명력을 2감소
190                 if player.life <= 0: #생명력이 0보다 작으면 게임을 종료하도록
191                     gameover = True
192         for b3 in bullets3: #총알3
193             if collision(player, b3):
194                 player.life -= 3 #생명력을 3감소
195                 if player.life <= 0: #생명력이 0보다 작으면 게임을 종료하도록
196                     gameover = True
```

- 총알1은 생명력을 1만큼 감소시키고, 총알2는 생명력2만큼, 총알3은 생명력3만큼 감소시키도록 구현하였습니다.

미션9: 사용자의 가장 오래 버틴 생존 시간을 최대 10개까지 파일에 기록한다.

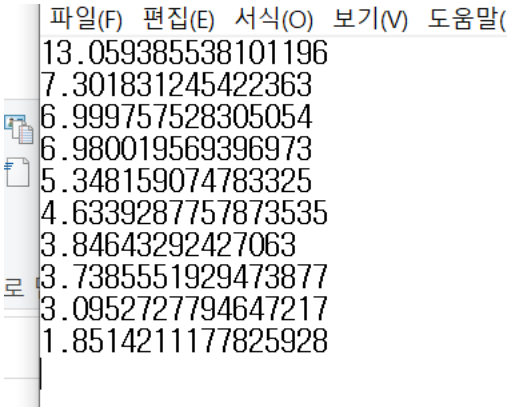
```
169 scoreboard = open("writescore.txt", "r")
170 scorelist = scoreboard.readlines()   #기존에 존재하는 writescore라는 텍스트 파일을 읽은 후 작업진행
171 if len(scorelist) < 10:
172     scoreboard = open("writescore.txt", "a") #정보가 쌓이도록
173     scoreboard.write(str(score) + "\n") #writescore.txt에 점수를 저장
174     scoreboard = open("writescore.txt", "r")
175     scorelist = scoreboard.readlines()  #텍스트파일에 저장된 점수들을 읽고 리스트로 가져와
176     scorelist = sorted(map(float, scorelist)) #정렬한다
177     scorelist = sorted(reverse = True) #가장 오래 생존한 시간부터 출력되도록
178     scoreboard = open("writescore.txt", "w") #기존의 내용들을 덮고 정렬된 점수로 표현시킨다
179     for item in scorelist:
180         scoreboard.write(str(item) + "\n")
181 elif len(scorelist) >= 10: #가장 길게 생존한 점수를 10개까지 저장하기 위해
182     scoreboard = open("writescore.txt", "a")
183     scoreboard.write(str(score) + "\n")
184     scoreboard = open("writescore.txt", "r")
185     scorelist = scoreboard.readlines()
186     scorelist = sorted(map(float, scorelist))
187     scorelist.sort(reverse=True)
188     del scorelist[10] #가장 적은 생존시간을 삭제 후
189     scoreboard = open("writescore.txt", "w") #최대 생존 점수 10개를 출력
190     for item in scorelist:
191         scoreboard.write(str(item) + "\n")
192
```



이름	수정한 날짜	유형	크기
boom.png	2020-06-09 오전 11:50	PNG 파일	7KB
bullet.py	2020-06-14 오전 12:05	Python File	2KB
collision.wav	2020-06-09 오전 11:12	WAV 파일	13KB
edit.py	2020-06-15 오후 3:50	Python File	8KB
fire.jpg	2020-06-13 오후 11:30	JPG 파일	99KB
healthgreen.PNG	2020-06-13 오후 9:39	PNG 파일	2KB
healthred.PNG	2020-06-13 오후 9:40	PNG 파일	2KB
heart.png	2020-06-13 오후 9:28	PNG 파일	13KB
light.jpg	2020-06-13 오후 5:20	JPG 파일	6KB
main.py	2020-06-14 오전 12:02	Python File	7KB
player.png	2020-06-01 오후 11:29	PNG 파일	2KB
player.py	2020-06-13 오후 11:56	Python File	2KB
text.py	2020-06-01 오후 11:29	Python File	1KB
writescore.txt	2020-06-15 오후 3:51	텍스트 문서	1KB

선택함 194바이트

- 코드는 다음과 같습니다. 2번째 사진에서 보이는 writescore.txt 파일에 기록들을 저장하도록 하였습니다. 최대 10개를 저장해야하기 때문에 텍스트 목록을 읽어온 후, 10개보다 적은 점수가 기록되어 있을 때와, 10개 이상의 점수가 기록되어 있을 때로 구분하여 최대 10개의 점수만 저장할 수 있는 코드를 작성하였습니다. 그리고 open 명령을 통해 파일을 읽고 쓸 때, 그 안의 내용들이 문자열로 읽어지고 써집니다. 만약 float형태로 정렬하지 않으면 10초의 기록이 2초의 기록보다 낮은 점수로 기록되는 문제가 발생합니다. 따라서 176,186줄의 코드와 같이 리스트들을 float의 형태로 받아 정렬하도록 하였고, 다시 출력 할때는 문자열의 형태로 적어야 하므로 180,191줄의 코드에서처럼 다시 string으로 변환 후 입력되도록하였습니다. 또한 scorelist.sort(reverse=True)를 통해 높은 점수부터 출력되도록 하였습니다. 그리고 178,189줄의 코드에서 "w"(write)모드로 파일 쓰기 모드를 설정하였는데 만약 "a"모드로 설정하면 같은 10개의 기록이 2번 출력되어 20개의 기록이 출력되는 문제가 생깁니다. 따라서 실행할 때마다 이전의 기록을 덮어쓰고 갱신하기 위해 "w"모드로 수행하였습니다. 다음 3개의 사진은 실행결과의 사진입니다.



파일(F) 편집(E) 서식(O) 보기(V) 도움말(

13.059385538101196
7.301831245422363
6.999757528305054
6.980019569396973
5.348159074783325
4.6339287757873535
3.84643292427063
3.7385551929473877
3.0952727794647217
1.8514211177825928

writescore.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

13.059385538101196
7.301831245422363
6.999757528305054
6.980019569396973
5.687870979309082
5.348159074783325
4.6339287757873535
3.84643292427063
3.7385551929473877
3.0952727794647217

- 실행결과의 사진입니다. 기존에 10개의 기록이 존재하였고, 이 상태에서 게임을 실행하고 5.7초의 기록을 얻었습니다. 다시 점수표에 들어가보면 기존의 가장 낮은 점수가 제거되고 새로운 기록이 갱신되는 것을 볼 수 있습니다.

미션10: 게임오버시에 기록된 생존시간을 화면에 출력한다.

```
130     if gameover:
131         draw_text("GAME OVER", 100, (WIDTH/2 - 300, HEIGHT/2 - 50), (255,255,255))
132         txt = "Time: {:.1f}  Bullets: {}".format(score, len(bullets1)+ len(bullets2) + 1)#총알1+총알2+총알3
133         draw_text(txt, 32, (WIDTH/2 - 150, HEIGHT/2 + 50), (255,255,255))
134         scoreboard = open("writescore.txt", "r")
135         scorelist = scoreboard.readlines() #이전 기록들을 가져와 읽기
136         tmp = 0 #기록들을 한줄마다 입력하기 위한 상수
137         for s in scorelist:
138             tmp += 1
139             draw_text("({}) : {}".format(tmp, s.strip()), 20, (50, 30*tmp), (255,255,255)) #30*tmp를 하여 각 리스트를 출력할때마다 각각 다른
```

#30*tmp를 하여 각 리스트를 출력할때마다 각각 다른 줄에 입력되도록

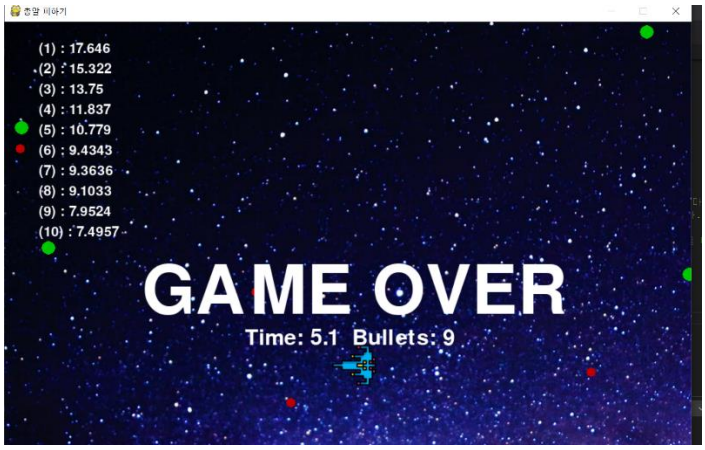
- gameover시에 출력하도록 하는 것이므로 if gameover 내에 작성하였습니다. 추가된 코드는 134~139줄의 코드입니다. 134~135줄의 코드를 통해 우선 기존에 저장되어 있는 기록들을 불러와 리스트로 읽습니다. 그리고 for문을 통해 1줄마다 입력하도록 하였습니다. 각 줄에 입력하기 위해서 각 리스트에 접근할 때마다 tmp를 1씩 증가시키고 그것을 30*tmp하여 y좌표가 달라지도록하

여 각 줄에 입력되도록 하였습니다. 그리고 (1) : 7.58 (2) : 5.42 이런식으로 표현되도록 format함수를 적절히 활용하였습니다. 2번째 사진은 139줄의 코드의 주석입니다.

```
195     for item in scorelist:
196         scoreboard.write(str(item)[0:6] + "\n") #소수점 간결 표현 위해 6개만 출력되도록
```

```
206     for item in scorelist:
207         scoreboard.write(str(item)[0:6] + "\n")
```

+ 기존에 있는 점수를 불러오면 14.33163626246437347 이런식으로 소수점이 너무 길어져서 간결하게 표현하기 위해 점수표에 6개의 문자만 저장하도록 기존의 코드에서 [0:6]를 추가하여 수정하였습니다. 다음 사진은 미션 10을 수행한 후의 결과 사진입니다.

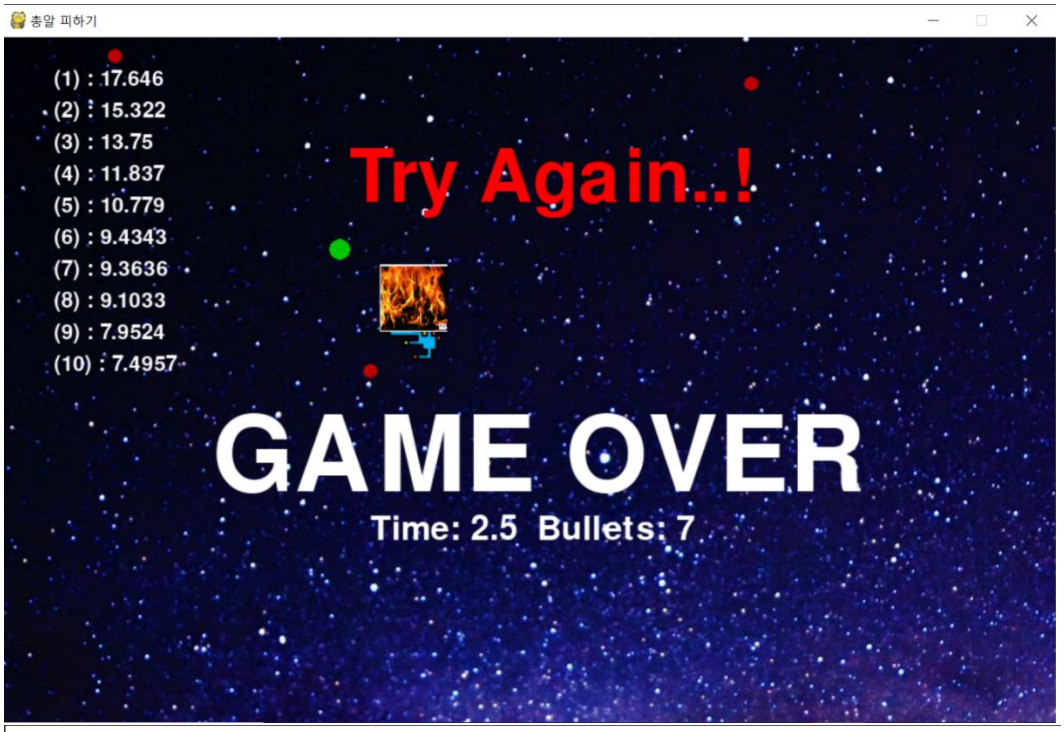


미션11: 게임오버시에 기록된 생존시간을 화면에 출력하며, 현재 기록이 순위권에 있을 경우 강조하여 표시한다.

```
130     if gameover:
131         draw_text("GAME OVER", 100, (WIDTH/2 - 300, HEIGHT/2 - 50), (255,255,255))
132         txt = "Time: {:.1f}  Bullets: {}".format(score, len(bullets1)+ len(bullets2) + 1)#총알1+총알2+총알3
133         draw_text(txt, 32, (WIDTH/2 - 150, HEIGHT/2 + 50), (255,255,255))
134         scoreboard = open("writescore.txt", "r")
135         scorelist = scoreboard.readlines() #이전 기록들을 가져와 읽기
136         tmp = 0 #기록들을 한줄마다 입력하기 위한 상수
137         for s in scorelist:
138             tmp += 1
139             draw_text("({}) : {}".format(tmp, s.strip()), 20, (50, 30*tmp), (255,255,255)) #30*tmp를 하여 각 리스트를 출력할때마다 각각 다른
140         if len(scorelist) < 10: #만약 점수가 10개보다 적게 저장되어 있으면 매기록이 10위 안에 드므로 무조건 ranking안에 존재한다.
141             draw_text("New record!! ->", 32, (100, 450), (0,255,0)) #시간점수 옆에 메시지를 띄어주고
142             draw_text("! Congratulations !", 70, (230, 100), (0,255,0)) #축하 메시지와
143             draw_text("It's in the ranking score!!", 32, (330, 500), (0,255,0)) #점수가 순위권 안에 있음을 강조한다.
144         elif len(scorelist) >= 10: #만약 점수가 10개이상이 되면, 점수표에 기록되어 있는 점수 중에서 가장 작은 수와 현재 스코어를 비교한다.
145             if score > float(scorelist[9]): #만약 현재 스코어가 순위권 안에 있다면
146                 draw_text("New record!! ->", 32, (100, 450), (0,255,0))
147                 draw_text("! Congratulations !", 70, (230, 100), (0,255,0))
148                 draw_text("It's in the ranking score!!", 32, (330, 500), (0,255,0))
149             elif score <= float(scorelist[9]):  #만약 현재 스코어가 순위권 안에 들지 못하는 점수라면
150                 draw_text("Try Again..!", 70, (330, 100), (255,0,0))    #다시 시도하여 순위권에 들라는 메시지를 보여준다
```

- 코드는 다음과 같이 작성하였습니다. 추가부분의 코드는 140~150줄의 코드입니다. 140~143줄의 코드는, 만약 점수표에 기록된 점수의 개수가 10개보다 작다면 매기록이 갱신기록입니다. 따라서

다음과 같이 작성하였습니다. 그리고 기록되어 있는 점수가 10개 이상일때를 144줄의 코드로 상황을 설정하였고, 145~148줄의 코드는 기록을 갱신했을 때의 코드입니다. 그리고 149~150줄의 코드는 순위권에 들지 못하였을 때의 코드입니다. 각각의 상황의 사진을 캡쳐하여 첨부하였습니다.



+ 마지막에 생명력을 5로 수정하여 제출하였습니다.